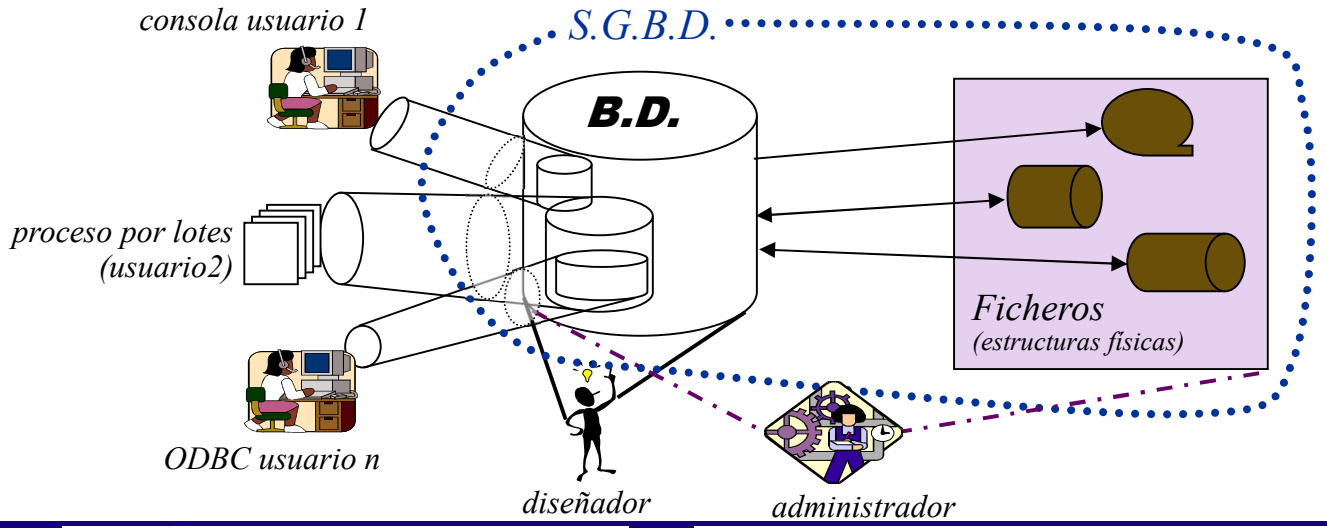
Tema 8: Sistemas Gestores de BBDD

- **Introducción. Aproximación cronológica.**
- **Arquitectura de un SGBDR: ORACLE®**
- **Administración de la BD ORACLE®**
  - Configuración y Gestión
  - Monitorización (estadísticas)
- **Afinamiento de la BD ORACLE®**
  - Estructuras: índices, clusters, parámetros
  - Procesos: planes de ejecución y hints
- **Conectividad de un SGBD (JDBC)**
- **Seguridad en SGBD. Precauciones.**
- **Concurrencia en un SGBD (Oracle®)**

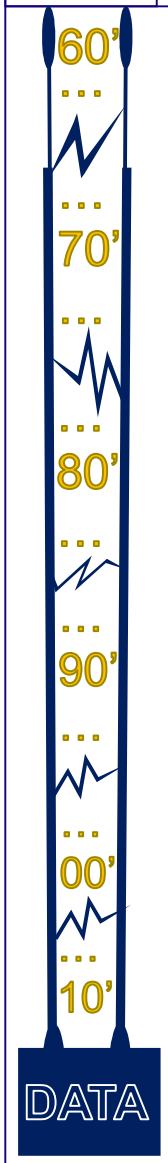
Tema 8: Introducción a SGBD (DBMS)

*Conjunto coordinado de **herramientas** que proporciona los medios necesarios para **interaccionar** con la base **a todos los niveles***

- herramientas: programas, procedimientos, lenguajes, ...
- interaccionar con la base: describir y manipular datos almacenados en la base, preservando su integridad, confidencialidad, y seguridad.
- a todos los niveles: usuario, programador, analista, ...



Tema 8: Aproximación Cronológica



Gestores **Navegacionales** (Jerárquico y en Red)
- apuntamientos físicos o relativos
- tecnología **eficiente** (OLTP, BD convencionales, …)

Gestores **Relacionales** (gestados en 70' / explotados desde 80')
- apuntamientos lógicos
- tecnología **eficaz** y accesible

Desktop Databases: adaptación de la tecnología a pequeña escala
Modelos Conceptuales: herramientas de diseño
Object Oriented databases: adaptación a un paradigma de análisis

Macrodatos y OLAP: interés y necesidad. Data Warehouses (DWH).

NoSQL (NoREL) – no structure, no model, no limit… BIG DATA
- nuevos enfoques (key-value, column, document, graph)
RDF y datos semánticos
CloudDB
NewSQL
OpenData
…

Next?

Tema 8: Algunos Eventos

1959	Nace el consorcio CODASYL
1964	Primer gestor: Integrated Data Storage (IDS; pre-network)
1966-68	IBM desarrolla el *Information Management System* (M. Jerárquico)
1968-69	Especificaciones y Lenguajes de Datos para los Gestores en Red Cincom lanza el SGBD TOTAL (shallow-network)
1970	Lanzamiento de ADABAS (online trans. proc. DB; inverted lists)
1971	…y otros similares como Datacom/DB (*key-driven*, for huge volumes)
1973…	Otros productos más cercanos al M. en red: IDMS
…	
1974	IBM desarrolla System R (gestor para el M. Relacional de EF Codd)
1978-79	Lanzamiento de **Oracle** V1-V2 (primer gestor relacional comercial)
1979-80	Lanzamiento de dBase (gestor para ordenadores domésticos)
1981	Le siguen **SQL/DS (IBM)**, Informix, Sybase, y otros gestores relacionales
1983	IBM lanza **DB/2**
…	
1990	Comercialización de OO-DBMS: Gemstone, Objectivity/DB, …
1992	SAP R3 (suite basada en la arq. *three-tier*)
…	
2004-06	BigTable… big boost for Big Data
2008	BigData spring…

Tema 8: Evolución Oracle (resumen)

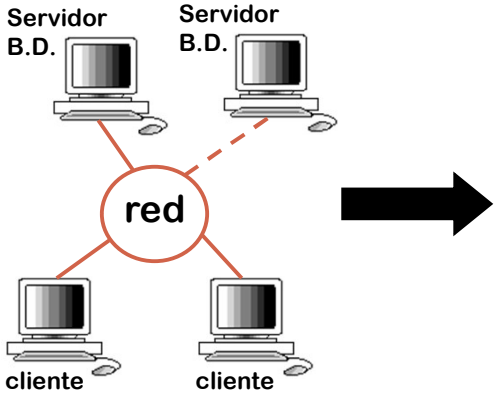
V. 1	1978	No llega a ser lanzada comercialmente
V. 2	1979	basic SQL operations (desarrollada en ensamblador)
V. 3	1983	desarrollada en C; g. concurrencia (transaccional) y distribución
V. 4	1984	incorpora consistencia (en lecturas) + ed. doméstica (para PC)
V. 5	1985	arquitectura cliente-servidor y BBDD distribuidas
V. 6	1988	PL/SQL + row-level locks + hot backup
V. 7	1992	int. referencial + procedures + triggers + motor ConText + optimizador
V. 8	1997	ORDBMS (7.3) or. objetos + multimedia storage + Oracle Spatial
V. 8i	1999	interoperabilidad con internet + Aurora (máquina virtual Java)
V. 9i	2001	RAC (Real Application Cluster) + XML support
V. 10g	2003	adaptada a *grid computing*
V. 11g	2007	mejorado en casi todo (eficiencia, seguridad, comodidad,…) + Exadata
V. 12c	2013	adaptada a *cloud computing* + in-memory engine + json + multitenant
V. *18c*	*2018*	*polymorphic table functions* (PTF) + más estándar

SystemR- -SQL/DS- ---------DB2----------
-Informix---------- -Sybase---- --Informix---(DB2)---------- -SQLServer-

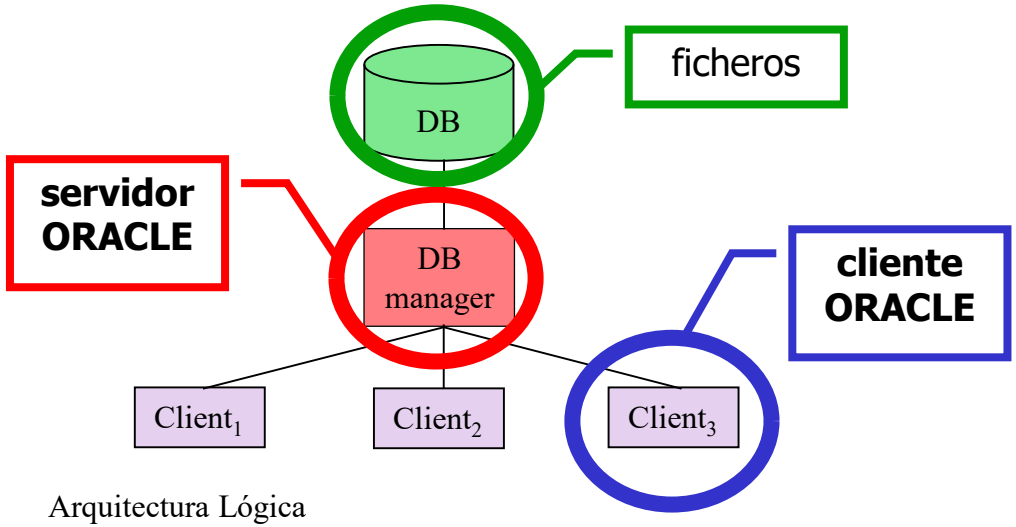
Tema 8.1: Ejemplo de SGBD

El SGBDR ORACLE®

- **ORACLE:** Sistema Gestor de Base de Datos Relacional;
  Versátil + probada Eficiencia y Escalabilidad + amplia Difusión
- Basado en el lenguaje de datos PL/SQL (extensión de SQL)
- Entorno multiusuario (Cliente/Servidor).
- El servidor alberga: repositorio Sw, ficheros, y servicios (instancias)



Arquitectura Comunicaciones



Arquitectura Lógica

Tema 8.1.1: Instancias ORACLE®

El **servidor Oracle** puede albergar varias instancias independientes (al menos una) que comparten repositorio Sw y se reparten los recursos)

Instancia:

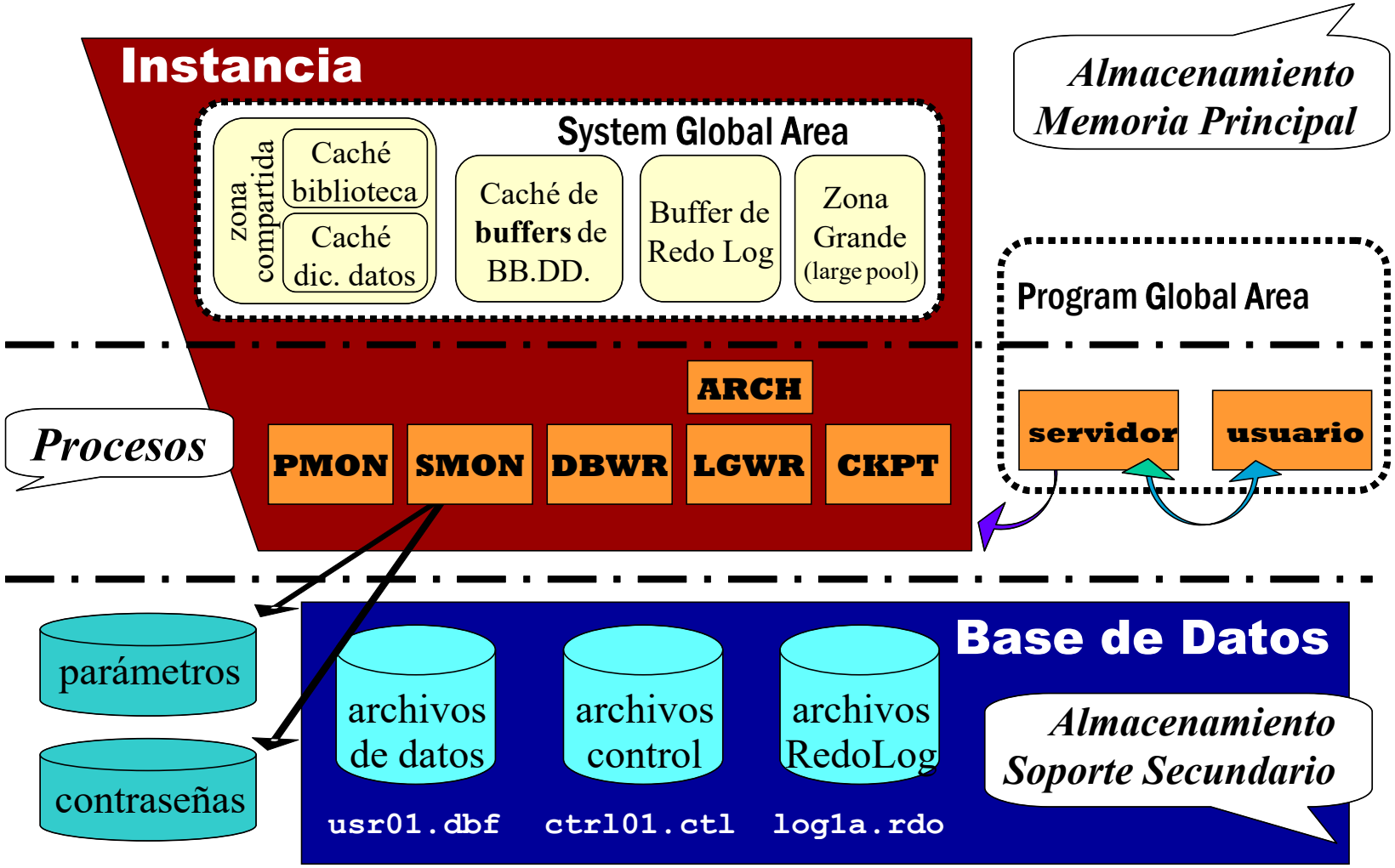
- Es un **conjunto completo de servicios** de BBDD
  (si existen o no varias instancias en el mismo servidor, es transparente)
- Servicios de acceso, control y uso de las BD.
- Se compone de **procesos** y **estructuras de datos** (físicas y en memoria)
- Sus recursos son compartidos por todos los usuarios (de la instancia).
- Las estructuras en memoria se organizan en dos áreas: **SGA** y **PGA**
- Las estructuras físicas se apoyan en el concepto de ***tablespace***.
  Cada tablespace podrá almacenarse en uno o más ficheros de datos.

Tema 8.1.1: Instancias ORACLE®

Instancias y Bases de Datos:

- Una **BD** es un conjunto de datos **almacenado** y **accesible** según una **estructura lógica** (*esq. relacional*, ~ cjto. de tablas interrelacionadas)
- Sus elementos pueden pertenecer a uno o más usuarios, almacenarse en uno o más *tablespaces*, pero siempre en **una sola instancia**.
- Dentro de una instancia, los objetos de una BD se referencian mediante la notación de punto (por ejemplo: *dueño.tabla.atributo*), si bien el usuario *dueño* puede omitir la primera sección.
- Un usuario (o cjto. de usuarios) aislado con sus objetos puede ser considerado como BD, aunque frecuentemente se asocia al concepto de BD todo lo contenido en la instancia.
- BBDD de distintas instancias pueden *federarse*, posibilitando su interrelación (si se requiere mayor integración → misma instancia).

Tema 8.1.1: Arquitectura ORACLE®



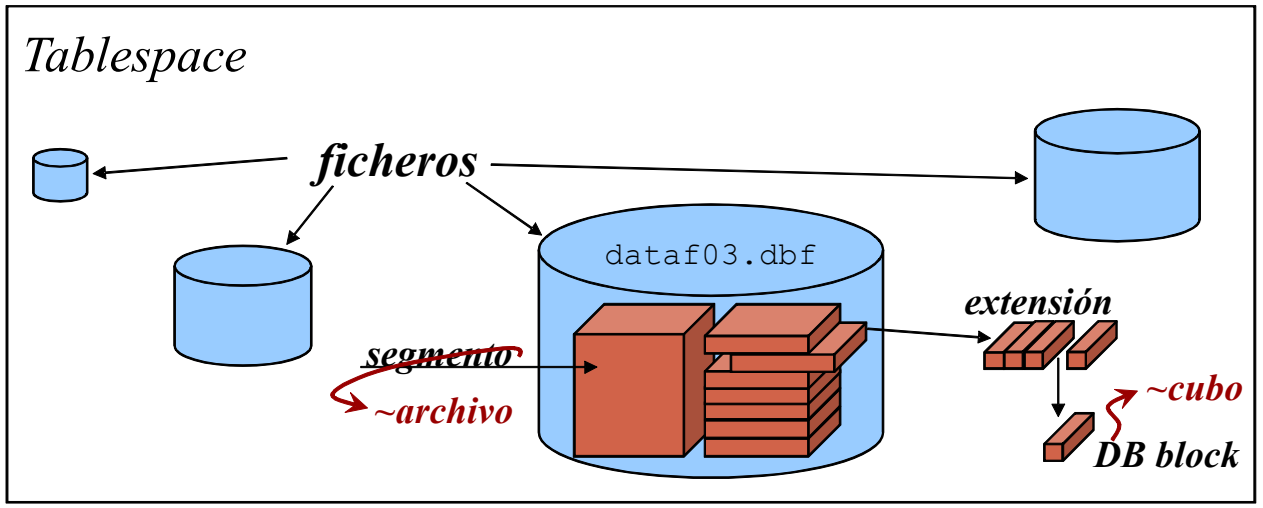
Tema 8.1.1: Ficheros de ORACLE®

- ***Parámetros*** (pfile / spfile): información para la inicializar la instancia.
- ***Contraseñas***: información de acceso a la instancia.
- ***Control***: contienen la información necesaria para la utilización de la instancia (nombre BD, nombre y ubicación de ficheros, back-up, etc.)
- ***Redo log***: protege la BD con *journaling* (registro de cambios que sufre la base, anotando cada operación antes y después de realizarse).
- **Datos** (*datafiles*): almacenan los segmentos de la BD

Tema 8.1.2: Esquema Interno ORACLE®

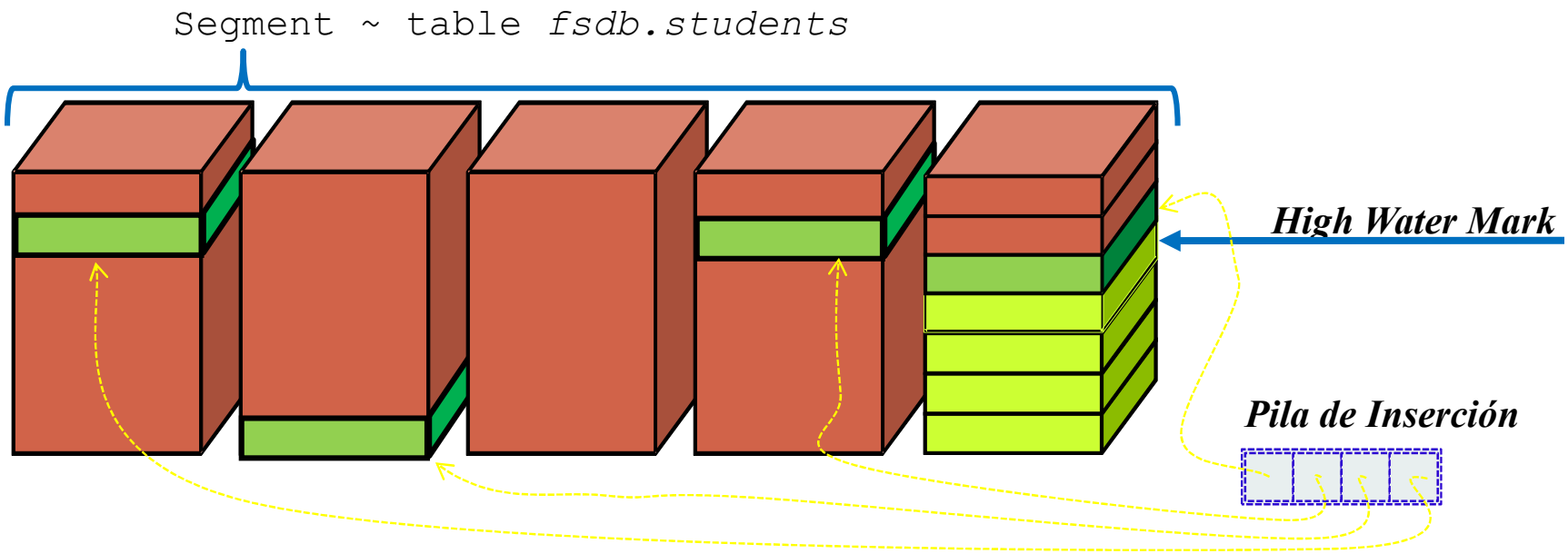
Tablespaces y Datafiles

- El ***tablespace*** es un almacén de datos.
- Puede tener asociados varios **ficheros** de datos (*datafiles*), y estos se asignan a un solo tablespace. El tamaño máximo de datafile es 32 GB.
- El tablespace se organiza en **segmentos**, uno para cada elemento (tabla, índice, ...) que contiene. Cada segmento se compone de **extensiones.**



Tema 8.1.2: Esquema Interno ORACLE®

Segmentos y Extensiones

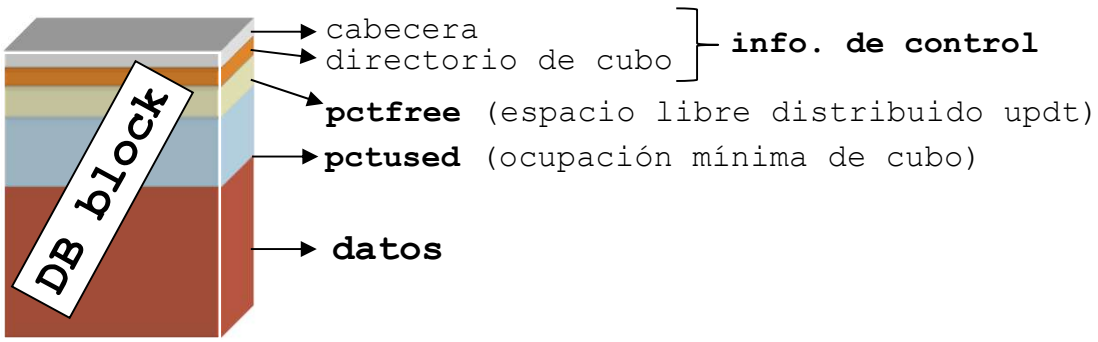


- Cada cubo (DBblock) es accedido conjuntamente (mediante el buffer).
- Los cubos de una extensión se almacenan y acceden de modo consecutivo, aprovechando (en su caso) la secuencialidad de disco.
- La operación TRUNCATE mueve el High Water Mark al primer bloque.

Tema 8.1.2: Esquema Interno ORACLE®

Cubos y Extensiones

- La **extensión** es un conjunto de *DB-blocks* contiguos asignados a un elemento
  - Cuando un elemento se crea, a su segmento se le asigna una extensión inicial
  - Cuando a un segmento se le acaba el espacio asignado, crece en una extensión
- El ***DB-block*** responde al concepto de **cubo** (1 'DB-block' ≡ 1 ó más bq físicos). Sus características y espacio (*blocksize*) son únicas para todo el tablespace, si bien pueden redefinirse para algunos objetos.



  - PCTFREE: porcentaje reservado para modificaciones (por defecto: 10%)
  - PCTUSED: porcentaje mínimo ocupado (por defecto: 60%); si una actualización deja al cubo demasiado vacío, éste será candidato para inserciones.

Tema 8.1.2: Esquema Interno ORACLE®

Cubos y Extensiones

- El *DB-block* admite cinco conf. de espacio: 2 KB, 4 KB, **8 KB**, 16 KB, y 32 KB.
- Un espacio de cubo grande
  - aprovecha la secuencialidad del dispositivo (disco),
  - aumenta la densidad (menos información de control y *gaps* más pequeños)
  - y reduce el tiempo de acceso a la totalidad (full-scan).
- Pero puede implicar
  - desperdicio de espacio (especialmente en clusters),
  - aumento de accesos a disco en procesos indexados,
  - menor eficiencia del *buffering*.
- Las extensiones grandes también aprovechan la secuencialidad de disco.
- El *DB-block* se corresponde con una página en el **buffer** (Mem $_{\text{Intermedia}}$).
  → debe tenerse un buffer distinto adaptado a cada espacio de cubo en uso.
- Determinados elementos (*cluster*) permiten el almacenamiento en **celdas** (*subconjunto del cubo*; su tamaño es divisor entero del cubo donde se aloja).
  Utilizar celdas puede mejorar la densidad, aprovechamiento de espacio, y coste de *fullscan*

Tema 8.2: Administración de BB.DD.

- La gestión de datos es el corazón de (casi toda) maquinaria empresarial. Garantizar su funcionamiento, seguridad y eficiencia es crucial.
- El administrador (DBA) es un profesional clave, con este perfil:
  - Conoce las tecnologías que soportan la BD (Hw, Sw, comm.)
  - Domina las tecnologías de BD (y los SGBD)
  - Conoce la estructura de sus BD (idealmente, involucrado en el desarrollo)
  - Otras competencias: sociales, organizativas, gestión personal, …
- Las funciones del administrador (DBA) incluyen:
  - **Gestión:** posibilitar el uso de la BD y administrar sus recursos
  - **Protección**: asegurar la persistencia y confidencialidad de la información
  - **Afinamiento: maximizar la eficiencia de la BD**
- Otros perfiles: científico de datos, ingeniero sistemas información, …

Tema 8.2.1: Configuración local (conn)

- Conexión: **sesión** establecida por un cliente para operar en una instancia
- Instancia: conjunto de procesos y estructuras (que albergan una BD)
- Datos de conexión: *configuración local* y *credenciales*
- Configuración local (cliente): se almacena en el fichero tnsnames.ora

```
SID_local =
   (DESCRIPTION = (ADDRESS_LIST = (ADDRESS =
                     (PROTOCOL=TCP)
                     (HOST=localhost)
                     (PORT=1521)
                  )                    )
                (CONNECT_DATA = (SERVICE_NAME = SID_host))
  )
```

Tema 8.2.1: Conexión

- Conexión: sesión establecida con un cliente para operar la BD.
- En la conexión se especifican las credenciales (usuario/password) y el identificador de instancia SID (o su descripción completa, en conexiones de tipo *thin*).
- La conexión podrá hacerse desde un lenguaje de programación (anfitrión) o mediante una aplicación cliente específica (**consola**).
- **sqlplus** (sql+): consola para conectarse con instancias BD Oracle

```
> sqlplus [username/password[@SID]] [AS role] [@script.ora] …
SQL> disconn[ect] /* commit & log out, without exiting */
SQL> conn[ect] [username[/password][@SID] [AS role] ]
SQL> exit          /* disconn and quit */
```

Tema 8.2.1: Gestión de Acceso

- Usuario: par name/passwd que da acceso a la BD según unos privilegios (ver diap. 4M9)
- Privilegios: acceso a operación de objetos de la BD, que pueden ser concedidos y revocados (ver diap. 4M.10)
- Rol: conjunto de privilegios; un usuario puede tener 0, 1 ó más

  Roles predefinidos: sysoper; sysdba (este puede operar la instancia):

```
SQL> conn sys/admin@ORCL AS sysdba
SQL> shutdown [{ABORT|IMMEDIATE|TRANSACTIONAL|NORMAL}]
SQL> startup [pfile=rutalocal] [force] [nomount] [quiet] …
SQL> disconn
SQL> conn / as sysdba
SQL> startup mount
```

Tema 8.2.2: Configuración de instancia

- La configuración se almacena en ficheros de parámetros (pfile/spfile)
  - pfile es textual (estático), y se edita con un editor de texto;
  - spfile es del servidor, y se actualiza con *alter system set variable=valor;*

  (scope={spfile|memory|both} indica si cambiar el fichero, el valor efectivo o ambos)
- La vista SYS.v$parameter contiene los parámetros efectivos
  (la fila name='spfile' indica si se está aplicando un pfile (null) o un spfile).
  También pueden visualizarse con la instrucción *show parameters* (sql*plus)
- Algunos par. se pueden cambiar en *caliente*, y otros req. reinicializar
- Algunos ejemplos de parámetros relevantes:
  - open_cursors = int (0..65535) default 50
  - spfile = ruta
  - db_block_size = int (2048..32768); default 8192
  - db_cache_size = int {k|m|g}
  - db_{2k|4k|8k|16k|32k}_cache_size = int {k|m|g}
  - job_queue_processes = int (0..1000)
  - compatible, log_archive_dest, cluster_database, sessions, …

http://docs.oracle.com/cd/B19306_01/server.102/b14237/initparams002.htm#CJAJHDED

Tema 8.2.3: Gestión del espacio

- **tablespace**: espacio en la base de datos para almacenar objetos. Tipos:
  - permanente: almacena los objetos persistentes
  - temporal: almacena objetos cuyo alcance no supera a la sesión
  - *undo*: almacena datos de recuperación (alternativa a segmentos de *rollback*)

```
CREATE [bigfile|smallfile] [TEMPORARY|UNDO] TABLESPACE <name>
       [{DATAFILE|TEMPFILE} <file_spec> [, <file_spec>…] ]
       [BLOCKSIZE <int> [k] ]
       [MINIMUM EXTENT <size> ] … ;
```

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7003.htm

- **datafile**: fichero de datos, asignado a un tablespace

```
ALTER  TABLESPACE <name>
       ADD DATAFILE <fichero> [SIZE <int> {k|m|g}];
```

http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_7003.htm

Tema 8.2.4: Monitorizar Instancias: *catálogo*

- el **catálogo relacional** en Oracle se denomina diccionario de datos:

```
SQL> SELECT * FROM DICTIONARY;

SQL> SELECT * FROM DICT_COLUMN; /* conviene seleccionar table_name… */
```

- Para simplificar el acceso, existen numerosas vistas (user/all/dba) que muestran objetos propiedad del usuario, accesibles por él, y de toda la BD
- Algunas de las vistas más utilizadas (no existen todas las combinaciones):

```
*_tables, *_tab_columns, *_views, *_constraints, *_source,
*_indexes, *_ind_columns, *_objects, *_catalog, *_synonyms,
*_tablespaces, *_users, *_role_privs, *_free_space, …
```

- Vistas de uso de espacio: `sm$ts_free, sm$ts_used, sm$ts_avail`
- Otras vistas interesantes (V$*):

```
v$session, v$process, v$rollstat, v$db_object_cache,
v$datafile, v$tablespace, v$database,…
```

Tema 8.2.4: Monitorizar Instancias

- Existen también diversas vistas que proporcionan **estadísticas de uso**. Algunas de las más utilizadas son las siguientes:
  - `v$sesstat`: estadísticas de las sesiones activas
  - `v$statname`: nombre de las estadísticas de la vista anterior
  - `v$sess_id`: operaciones i/o lógicas y físicas por cada sesión
  - `v$filestat`: lecturas/escrituras en cada datafile
  - `v$librarycache`: rendimiento de la caché de la sga
  - `v$sgastat`: estadísticas de sga global
  - `v$sqlarea`: estadísticas de la caché de cursor (workspaces)
- ***Autotrace*** proporciona el plan y algunas estadísticas: `set autotrace on`
- Por otro lado, existen estadísticas del optimizador (para aplicar opt. por coste), configurables con el paquete *dbms_stat*, y activadas (enable/disable) con el paq. *dbms_auto_task_admin* (parám. client_name ='auto optimizer stats collection')

Tema 8.3.1: Afinamiento (tuning)
Estructuras: Indexación

- La selección de índices (**ISP**) forma parte del *diseño físico*.
- Consiste en decidir las **estructuras auxiliares** para optimizar el rendimiento de la BD de acuerdo a los procesos que la actualizan o consultan
- La sintaxis básica de creación de indices en ORACLE® es:

```
CREATE [ind_type] INDEX ind_name
       ON table_name(ind_key) [FROM … ];
```

donde

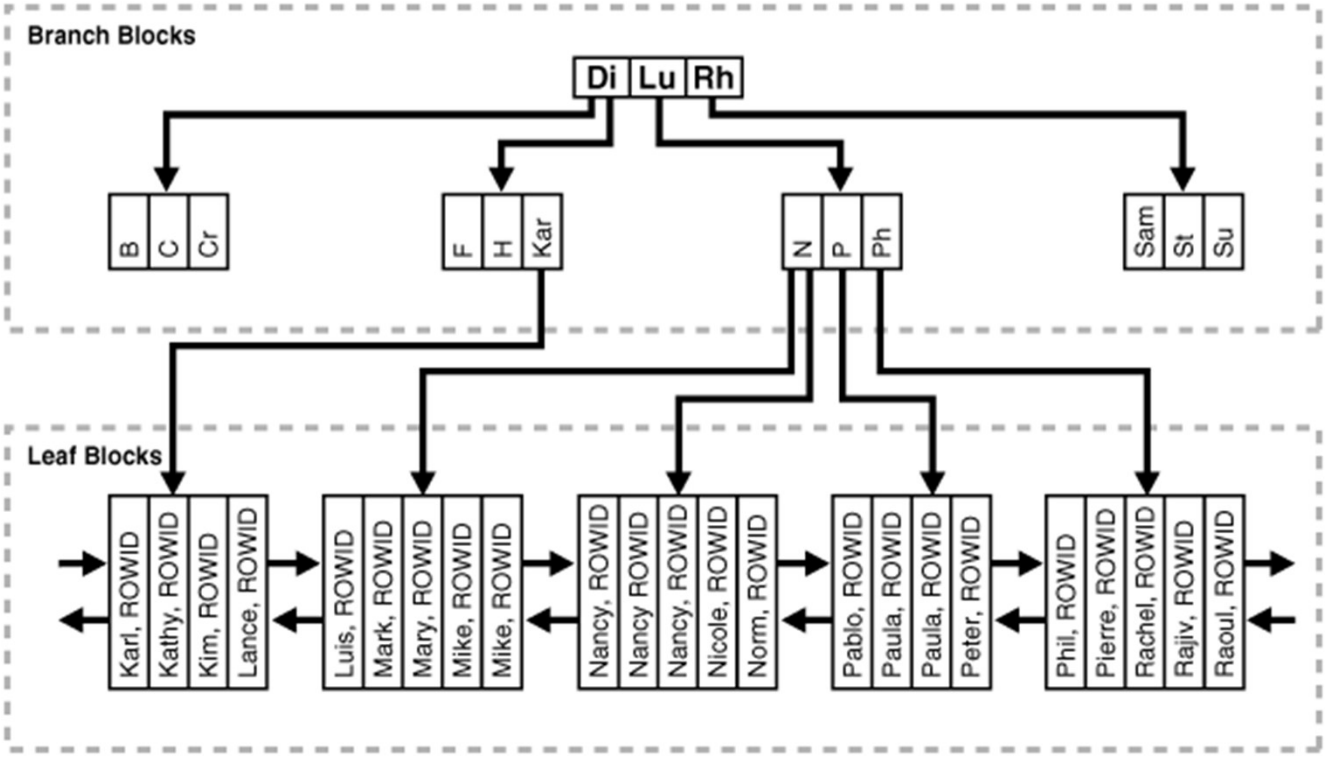
primario *bit map* *secundario*

```
ind_type := UNIQUE | BITMAP |    (default)
```

`ind_key :=` columna(s) de *table_name*, separadas por comas
(o una función sobre esas columnas...)

Tema 8.3.1: Afinamiento - Estructuras Indexación en Oracle®

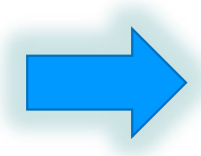
- Primario en B tree, secundario en B^+ tree:



Tema 8.3.1: Afinamiento - Estructuras Indexación en Oracle®

- Bitmap *clusterizado* Oracle®:
  - Debe aplicarse sobre objetos poco volátiles
    (tablas constantes o vistas materializadas reconstruidas periódicamente)
  - Comprime valores repetidos adyacentes
    (puede ser eficiente con *#valores*>1% si la clusterización es elevada)
  - Requiere un bit más por esquema (semántica: '*new bm*'/'*same as previous*')
  - Al tener tamaño reducido, mezclar índices es eficiente

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N	●	●	●							●	●	●	●			
S								●	●							
E				●	●	●	●									
W														●	●	●

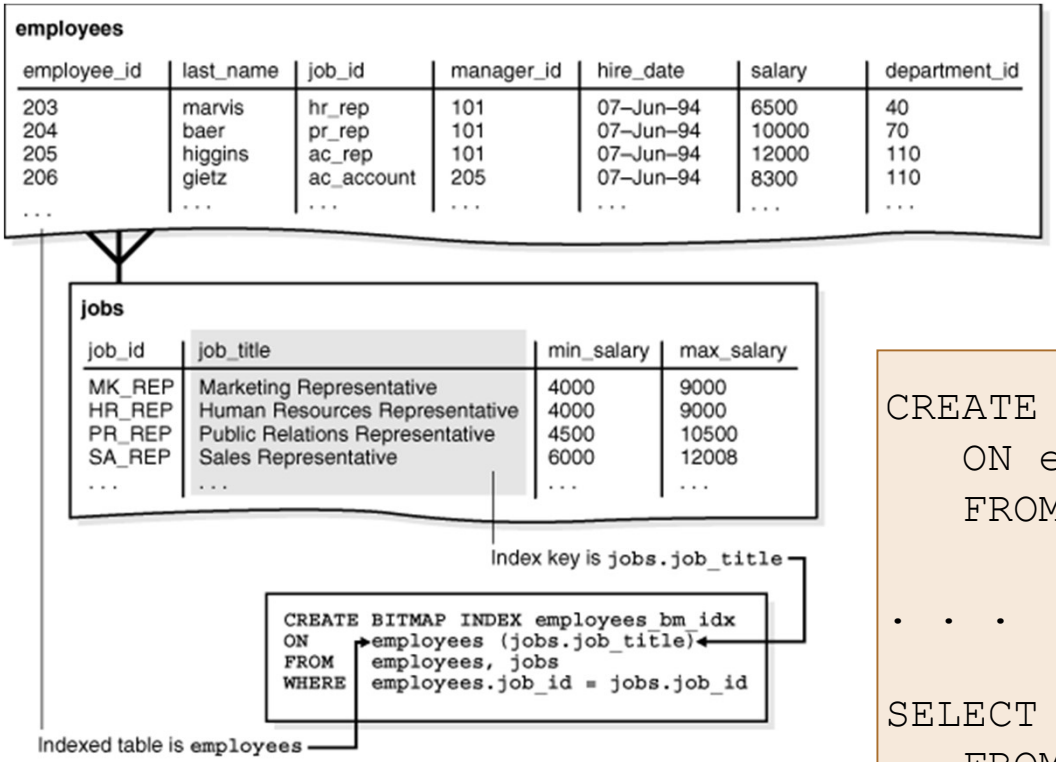


	1	4	8	10	14
N	●			●	
S			●		
E		●			
W					●

Tema 8.3.1: Afinamiento - Estructuras

Indexación en Oracle®

- Ejemplo de índice en Oracle®: índice sobre columna externa (*bitmap join index*)




```
CREATE BITMAP INDEX emp_jobs_idx
   ON employees (jobs.job_title)
   FROM employees JOIN jobs
        USING (job_id);
. . .

SELECT last_name, salary
   FROM employees NATURAL JOIN jobs
   WHERE jobs.job_title = 'Currito';
```

Tema 8.3.2: Afinamiento - Estructuras

Clusterización en Oracle®

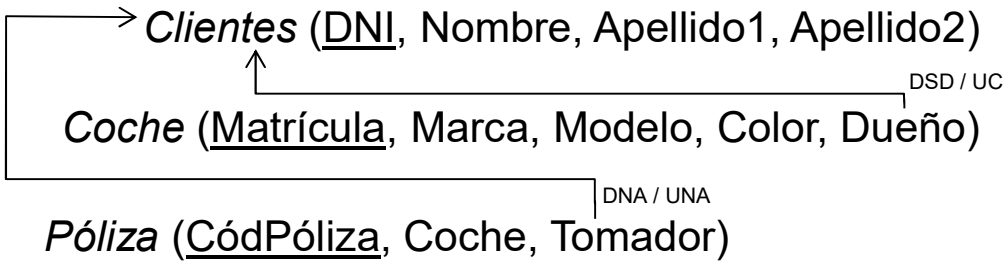
- Para ORACLE, un *cluster* es la definición de clave privilegiada.
- A través del cluster, varias tablas pueden almacenar físicamente los datos combinados mediante esa clave (eficiente para JOIN y accesos por la clave privilegiada, ineficiente para todo lo demás).
- El cluster debe **crearse antes de crear la tabla**
- El cluster garantiza que **toda la fila** combinada (el resultado del join de todas las tablas implicadas para un valor del cluster) se almacena físicamente **en el mismo cubo**

- **Ventaja:** el acceso a elementos combinados es más eficiente
- **Inconveniente:** el acceso individual puede ser muy ineficiente

Tema 8.3.2: Afinamiento - Estructuras

Clusterización en Oracle®

Ejemplo:




```
CREATE  CLUSTER identidad (DNI VARCHAR2(9));
CREATE  TABLE cliente(…) CLUSTER identidad (DNI);
CREATE  TABLE coche(…) CLUSTER identidad (dueño);
CREATE  TABLE poliza(…) CLUSTER identidad (tomador);
CREATE INDEX ind_dni ON CLUSTER identidad;
```

```
identidad
( DNI C(9),
  cliente (nombre C(25), apellido1 C(15), apellido2 C(15)),
  coche (matrícula C(7), marca C(20), modelo C(20),color C(10) )*,
  poliza (cod C(30), coche C(7) )*
);
```

Tema 8.3.2: Afinamiento - Estructuras

Clusterización en Oracle®

Ejemplo 2:

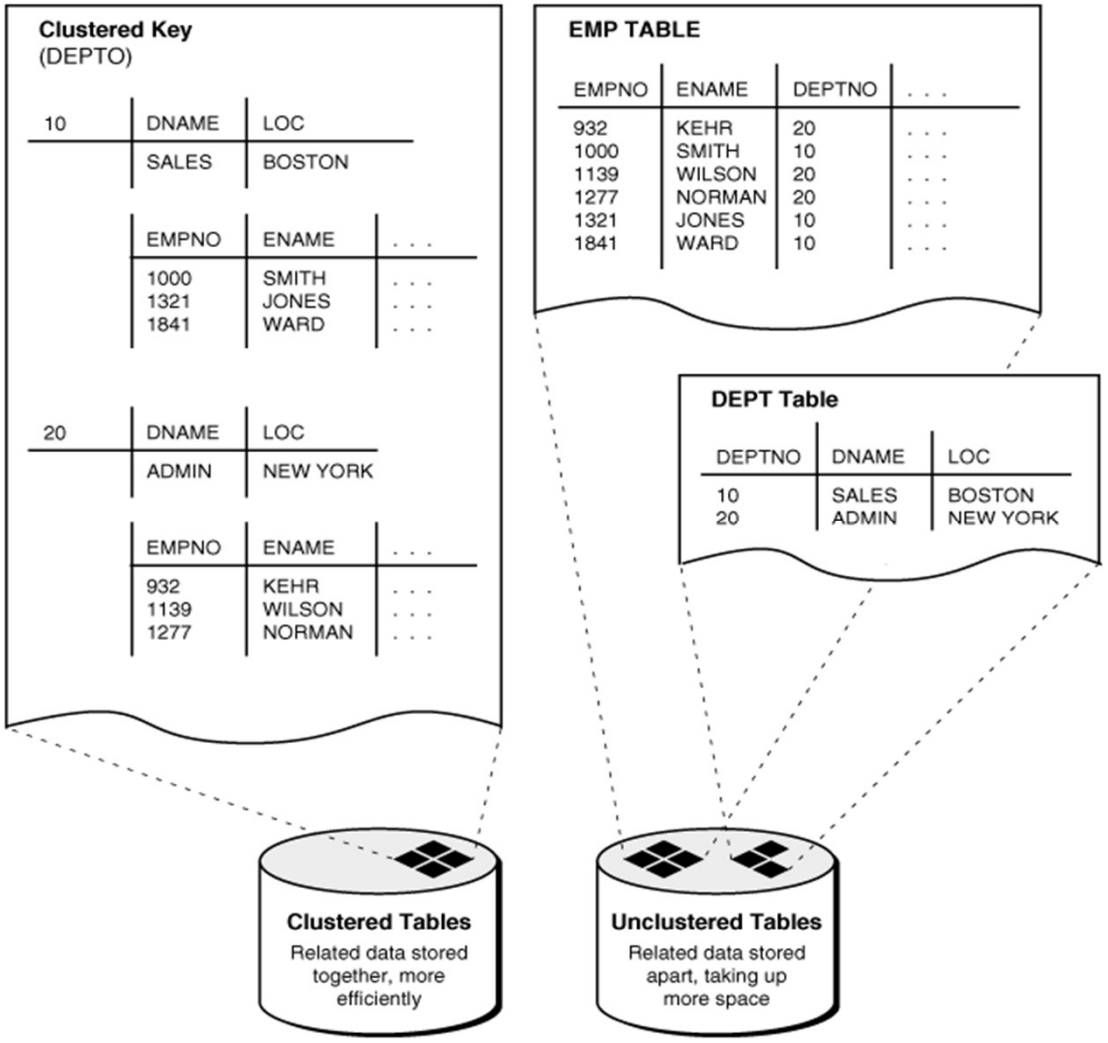
```
CREATE CLUSTER emp_dept (deptno NUMBER(3))
   SIZE 600
   TABLESPACE users
   STORAGE (INITIAL 200K NEXT 300K
            MINEXTENTS 2 PCTINCREASE 33);
```

```
. . .

CREATE TABLE dept
(deptno NUMBER(3) PRIMARY KEY,
 . . . )
CLUSTER emp_dept (deptno);


CREATE TABLE emp (
empno NUMBER(5) PRIMARY KEY,
ename VARCHAR2(15) NOT NULL,
. . . ,
deptno NUMBER(3) REFERENCES dept)
CLUSTER emp_dept (deptno);

. . .
```



Tema 8.3.2: Afinamiento - Estructuras

Clusterización en Oracle®

- La **elección de la clave** es **crítica**: puede bajar la densidad.
- Como otros objetos, permite definir características físicas
- El *cluster* puede ser **indizado** o **disperso** (con **orden** opcional).
- Un *cluster* ***mono-tabla***: permite cambiar la organización base
  - **Ventaja**: eficiencia en acceso por clave clusterización
  - **Inconveniente**: menor densidad, peor eficiencia en accesos por claves de selección no privilegiadas.

Tema 8.3.3: Afinamiento (tuning) Parámetros Físicos

- Oracle permite definir parámetros físicos en la creación de objetos (tablas, clusters, índices, vistas materializadas).
- **Tablespace**: al definir este parámetro, se pueden elegir otros como el *blocksize* (espacio de cubo) que puede ser distinto en cada tablespace (cuidado con definir cubos no adecuados). En la instancia, se pueden definir cachés de cada formato.
- En los *clusters*, SIZE permite definir *celdas* ('cubos' más pequeños que el cubo)
- Espacio libre distribuido: `PCTFREE` y `PCTUSED` a la medida del objeto
- **STORAGE**: parámetros de almacenamiento
  - Tamaño de las extensiones: `initial, next, pctincrease, maxsize, maxextents, minextents`
  - Memoria intermedia: cuál de los *pools* será utilizado para ese objeto: `buffer_pool {keep|recycle|default}`
  - … y algunos más (como las listas de puntos de inserción, o `freelists`)

Tema 8.3.4: Afinamiento (tuning)

Optimización de Consultas en Oracle®

- La mayoría de instrucciones de manipulación de datos pueden resolverse, en general, siguiendo distintos caminos físicos (distintos algoritmos).
- Es importante describir bien las consultas (algebraicamente) para conseguir ejecuciones más directas y eficientes.
- Si bien es cierto que muchos SGBD tienen un Optimizador capaz de *reinterpretar expresiones* (para operar $\sigma_{=} \times$ como si fuera $*$, por ejemplo) es conveniente describirlas bien para no depender de ese componente.
- Además, es necesario conocer los detalles de aplicación de cada operador en el SGBD en uso, para controlar los mecanismos que se utilicen:
  - Ejemplos en Oracle:
    - el operador LIKE fuerza un recorrido serial: `... where name LIKE 'John'`
    - una función/operación evita el uso de índices: `... where enddate+0 <...`

Tema 8.3.4: Afinamiento (tuning)

Plan de Ejecución

- El camino físico para resolver una instrucción de manipulación de datos, también denominado *plan de ejecución*, es el resultado de una o varias secuencias de decisiones, que se pueden representar de forma arbórea (árbol de decisión).
- El SGBD suele contemplar instrucciones (en el LCD) que permiten al usuario conocer el plan de ejecución.
- Para elegir el camino físico que resolverá la selección descrita, se pueden seguir distintas estrategias. El SGBD contará con una o más de estas estrategias (y mecanismos para determinar cuál seguir, si son varias).
- Hay que establecer un objetivo: priorizar reducir el coste de los primeros resultados o el coste de la operación completa. Depende si el consumidor aprovecha los resultados según se obtienen o necesita la completitud.

Tema 8.3.4: Afinamiento (tuning)

Obteniendo el *Plan de Ejecución*

- La instrucción SQL para obtener un plan de ejecución es **EXPLAIN**

```
EXPLAIN PLAN [SET statement_id = '...'] FOR <dml_sentence>;
```

  - EXPLAIN requiere privilegios de consulta en … **V_$SESSION, V_$SQL, V_$SQL_PLAN, V_$SQL_PLAN_STATISTICS_ALL**

- En Oracle, el plan se almacena en una tabla temporal global (de sesión) denominada **SYS.PLAN_TABLE$** (con sinónimo **PLAN_TABLE**)
  - La tabla de planes se crea automáticamente (versión 10). En versiones anteriores (o para crear una tabla local permanente) utiliza utlxplan.sql

- EXPLAIN proporciona una previsión de plan 'a priori' (sin ejecutarlo)
- En Oracle, el modo autotrace (**set autotrace on**) proporciona una descripción básica del plan efectuado, y un resumen de estadísticas.

Tema 8.3.4: Afinamiento (tuning)

Consultando el *Plan de Ejecución*

- Se puede consultar un plan directamente de **PLAN_TABLE** …

```
SELECT * FROM PLAN_TABLE WHERE statement_id='…';
```

- …pero es una tortura interpretar sus 36 ilegibles columnas, especialmente porque cada plan consta de varias filas.
- Para simplificarlo, Oracle facilita el paq. **DBMS_XPLAN** con la función **DISPLAY(plan_table, statement_id, format, filter)**

```
SELECT * FROM TABLE(DBMS_XPLAN.DISPLAY);

SELECT PLAN_TABLE_OUTPUT FROM TABLE
   (DBMS_XPLAN.DISPLAY(NULL,'statement_id','BASIC'));

SELECT * FROM TABLE (DBMS_XPLAN.DISPLAY(FORMAT=>'+ALLSTATS'));
```

- Formatos: **BASIC | TYPICAL | ALL** (se pueden añadir más opciones)
  https://docs.oracle.com/database/121/ARPLS/d_xplan.htm#ARPLS74741

Tema 8.3.5: Afinamiento (tuning)

Optimización basada en… *Reglas* vs *Costes*

- El plan de ejecución se puede alcanzar por distintos métodos, cuya disponibilidad dependerá del SGBD. Los más habituales son :
  - REGLAS: cada decisión se toma en base a criterios apriorísticos, que presentan buen comportamiento (en general).
  - **COSTES**: las decisiones se toman calculando la mejor opción; el camino será óptimo si los datos (*estadísticas de estado de la BD*) son fiables. Esa fiabilidad dependerá de la significatividad del muestreo elegido, de la frecuencia de actualización, etc.
  - ESTADÍSTICAS DE USO: algunos gestores almacenan estadísticas de uso (aplicación de planes pasados), permitiendo adoptar aquellas decisiones que anteriormente fueron exitosas.
  - DIRIGIDOS por el usuario: el usuario interviene en el proceso de decisión. Para esta vía, Oracle cuenta con el mecanismo de *Hints*.

Tema 8.3.5: Afinamiento (tuning)

HINTS en Oracle®

- Oracle puede que no elija un camino óptimo
  (no utiliza los indices creados o los utiliza no de manera eficiente)
- Los HINTS fuerzan el camino físico para resolver sentencias (select, insert, delete, update), son especificados como comentarios

```
SELECT /*+ HINT */ attributes FROM tablename … ;

SELECT --+ HINT
   attributes FROM tablename … ;
```

- Se pueden especificar varios HINTS para la misma instrucción (separados por espacios).
- Muchos de los HINTS en Oracle cuentan con una versión opuesta, con la misma sintaxis precedida de NO (ejemplo: INDEX→ NOINDEX)

Tema 8.3.5: HINTs Oracle® más usuales

Sintaxis del HINT			Descripción
`/*+RULE */`	`/*+ALL_ROWS*/`	`/*+FIRST_ROWS(n)*/`	elige tipo de optimizador (reglas/costes)
`/*+ FULL(tablename) */`			recorrido a totalidad (*full scan*) de tabla
`/*+ ROWID(tablename) */`			rowid scan (cubos individuales)
`/*+ CLUSTER(tablename) */`			tabla en cluster accedida por el mismo
`/*+ HASH(tablename) */`			usa la dispersión de una tabla en *cluster*
`/*+ ORDERED */`	`/*+ LEADING(tab1 tab2)*/`		join tables in Q order, or specify an order
`/*+USE_NL(t)*/`	`/*+USE_MERGE(t)*/`	`/*+USE_HASH(t)*/`	join tables with nested loops/merge/hash (*inner*)
`/*+ INDEX(tablename) */`	`/*+NO_INDEX(table)*/`		use/forbids any index on the given table
`/*+ INDEX(tablename index1 index2 ... ) */`			use/forbids specific index/es (one or +)
`/*+ AND_EQUAL(tablename index1 index2 […]) */`			use more than one index (up to 5)
`/*+ INDEX_ASC(…) */`	`/*+ INDEX_DESC(…) */`		index range scan in asc/desc order
`/*+ INDEX_FFS(…) */`	`/*+ INDEX_SS(…) */`		index fast full scan // index skip scan
`/*+ INDEX_JOIN(tablename [indexname […]]) */`			join indexes (sort of inverted access)
`/* CACHE */`	`/* NOCACHE */`		situar cubos al comienzo/final de la lista LRU
`insert /*+ APPEND */ …`	`insert /*+NOAPPEND*/`		escritura directa HWM (~~buffer, stack, RI, trigger~~)

Tema 8.4: Conectividad con BB.DD.

- Para el desarrollo de aplicaciones complejas, existen lenguajes de prog. *anfitriones* capaces de conectar y operar BD relacionales: Pro*C, Java…
- En Java se usa la API JDBC (Java DataBase Connectivity)
- El paquete java.sql proporciona las clases que lo implementan
  http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-10201-088211.html

  DriverManager, SQLException, Connection, Statement, ResultSet,…

- Para usar BD Oracle, es necesario contar además con el manejador jdbc para Oracle (que debe registrarse) y las clases de Oracle SQL:

```
import java.sql.*;
import oracle.jdbc.driver.*;
import oracle.sql.*;
DriverManager.registerDriver(
                    new oracle.jdbc.driver.OracleDriver());
```

Tema 8.4: Conexión por JDBC

- El objeto *conexión* establece un canal de comunicación para enviar instrucciones SQL y obtener el resultado
- Puede hacerse con el driver oci de Oracle, utilizando la definición local de instancias (tnsnames)…

```
Connection conexion = DriverManager.getConnection(
   "jdbc:oracle:oci8:@sid", "username", "password");
```

- … o bien mediante el driver *thin* (definición explícita)

```
Connection conxn2= DriverManager.getConnection(
   "jdbc:oracle:thin:IP:puerto:srv","username","password");
```

- Conviene imbuirlo en *try {...} catch {...}* para manejar excepciones (SQLException), por si el servidor rechaza la conexión, está caído, …
- Al finalizar, debe cerrarse el objeto: `conexion.close();`

Tema 8.4: Instrucciones y Resultados

- A través de una conexión, se puede instanciar una instrucción…

```
Statement instruccion = conexion.createStatement();
```

- …que podemos ejecutar y de la que podemos obtener su resultado:

```
ResultSet resultado = instruccion.executeQuery(
            "select * from dual");
```

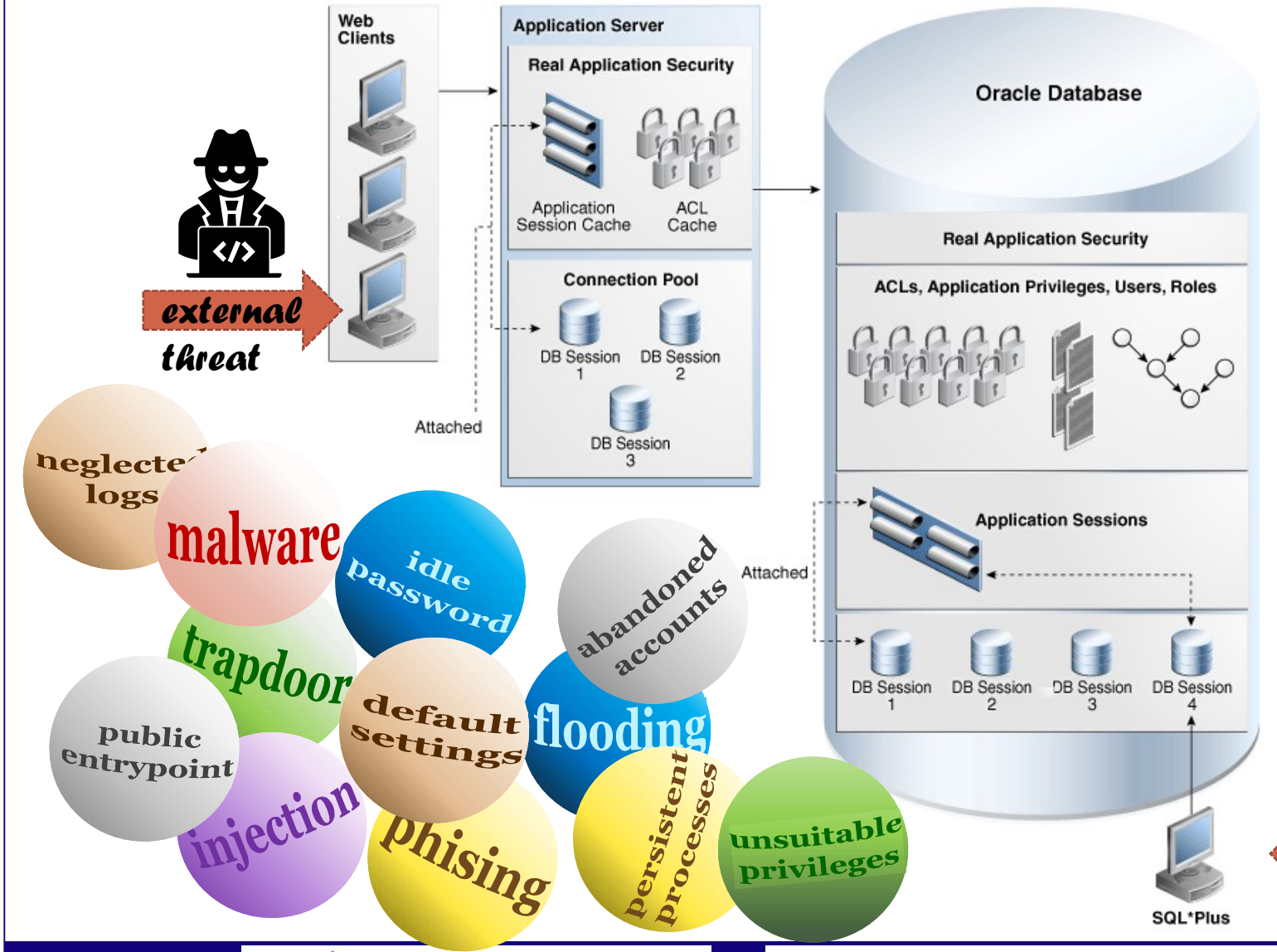
- Ese objeto *ResultSet* puede consultarse y actualizarse
  http://docs.oracle.com/javase/7/docs/api/java/sql/ResultSet.html
- ResultSet se recorrer fila a fila (mantiene un puntero actualizable por *first, last, relative(int), next, previous*), y de cada fila se puede obtener cualquier columna:

```
While resultado.next()
      {System.out.print(resultado.getString(1));}
```

- Al finalizar, deben cerrarse los objetos:

```
resultado.close();
instruccion.close();
```

Tema 8.5: Seguridad y Riesgos



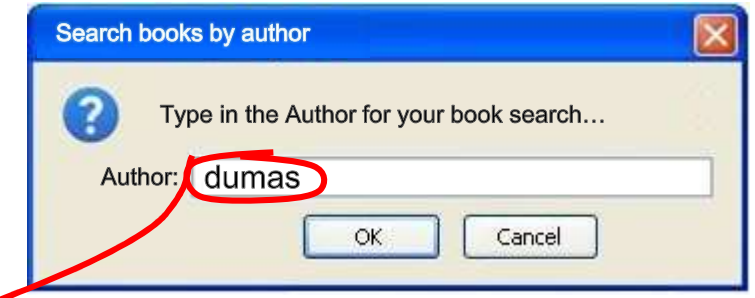
GOOD HABITS

- users training
- Hygiene
- Audit
- Secure Sw
- Mistrust

Tema 8.5: Precauciones - Inyección

- Alguien malintencionado podría alterar el resultado de una consulta.

```
…"SELECT title from books
    where author = '"||txt||"';"…
```



- ¿Y si el "autor" que busco fuera…?

```
' UNION SELECT table_name FROM user_tables; --
```

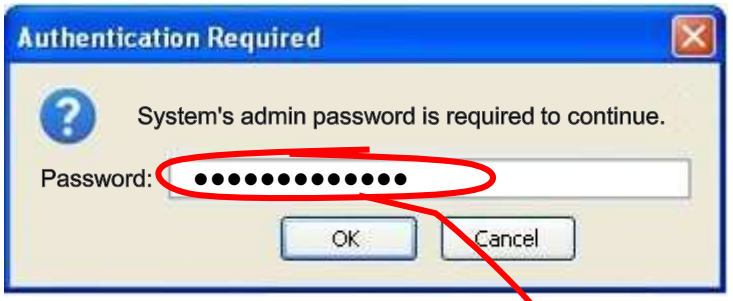
- En ese primer paso veo que existe una tabla 'orders', y ahora "busco"…

```
'; DROP TABLE orders; --
```

- ***Precauciones***:
- protege tus **metadatos** y **estructuras** controlando los privilegios.
- Crea usuarios específicos para cada aplicación y otorga **privilegios** mínimos.
- Crea **vistas** para consultas, y tablas **to_do** para escrituras (el usuario registra la operación a realizar, y otro proceso lee la tabla, analiza la acción y la hace si procede).

Tema 8.5: Precauciones - Inyección

- Extraer texto libre de un formulario y concatenarlo a una instrucción es muy peligroso, porque el usuario puede escribir lo que quiera…




```
ResultSet resultado = instruccion.executeQuery(
 "SELECT sueldo FROM nominas
    WHERE EXISTS (SELECT 'x' FROM credentials
                  WHERE usr='SYS' AND (passw='"+txt+"');"
);
```

- En este ejemplo, ¿Qué pasaría si el usuario escribe…? `' or 'x'='x`

- ***Precauciones***: procesa las cadenas de texto procedentes de formularios, asegurando la literalidad de caracteres de control (escape):

```
searchWord.replace("\\","\\\\").replace("'","\\'")
```

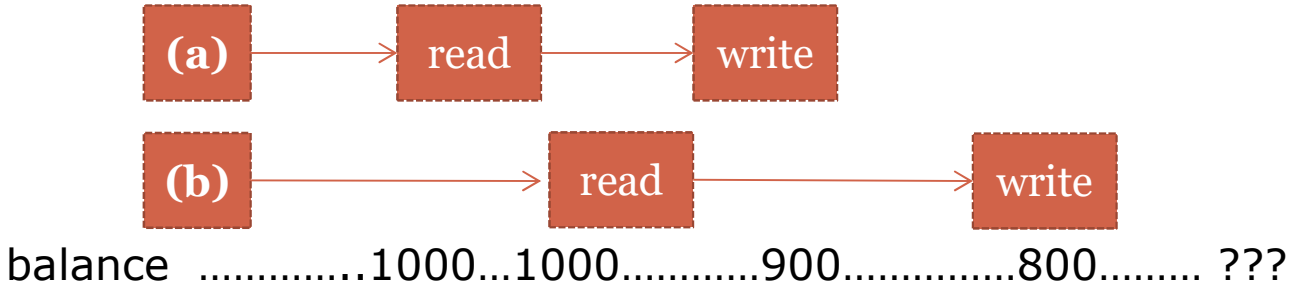
Tema 8.5: Recuerda a *Bobby Tables*

Tema 8.6: Concurrencia - Definición

- **Concurrencia**: varios usuarios operan la BD simultáneamente.
- Condición de Carrera: si una operación requiere varios pasos afectando alguno a un recurso compartido (en este caso, el estado de la BD), y el resultado depende de la secuencia que está afectada por otro proceso que opera el mismo recurso.

  Ejemplo: proc. a) `UPDATE account SET balance=balance-100;`
  proc. b) `UPDATE account SET balance=balance-200;`

  Ambos procesos requieren leer el estado (p.e., balance=1000) y después escribir el resultado; si ambos leen a la vez y luego escriben su resultado parcial, el resultado final podría ser erróneo (800 ó 900, en lugar de 700).



Tema 8.6: Concurrencia - Transacción

- En BD, las carreras pueden afectar incluso a la integridad de los datos.

Ejemplos:

falta de entidad

```
a) INSERT INTO client (DNI,name) VALUES (123, 'John');
b) INSERT INTO client (DNI,name) VALUES (123, 'Mary');
```

falta de integridad

```
a) DELETE client WHERE DNI=123;
b) INSERT INTO car (plate,owner) VALUES (0000AEI, 123);
```

lectura sucia

```
a) SELECT saldo INTO var FROM accounts WHERE DNI=123;
b) UPDATE accounts SET saldo=saldo-100 WHERE DNI=123;
```

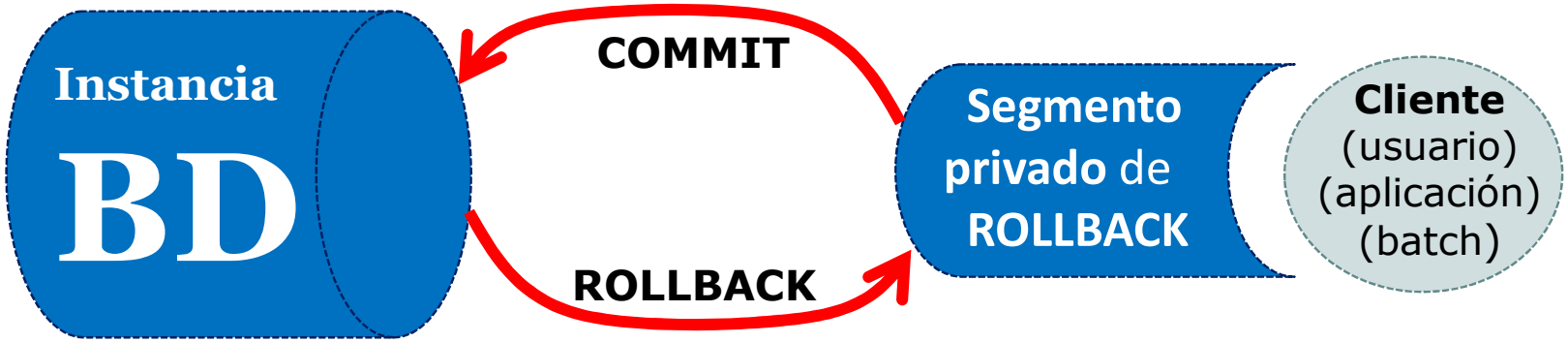
- Para resolverlo, se define la *granularidad* al nivel de *transacción*.
  - granularidad de ejecución: especificidad con la que se definen las ejecuciones atómicas (cuya secuencia no puede ser interrumpida).
  - transacción: secuencia de operaciones que se opera en conjunto (indivisible o atómica), pudiendo ser perpetrada o desestimada. Comienza con la primera instrucción DML (o con *set transaction*) y finaliza al perpetrarse o desestimarse (también con el *fin de sesión*).
  - granularidad de bloqueo: esquema, tabla, columna, fila, …

Tema 8.6: Concurrencia - Multicopia

- **Multicopia**: para aliviar la gestión de la concurrencia en el servidor, se puede mantener una copia (virtual) de la BD para cada sesión abierta.
  - Las actualizaciones no perpetradas que se operen en una sesión es almacenado en un segmento de **rollback**.
  - El estado de la BD visible desde cierta sesión es el estado general más las operaciones en el segmento de rollback.
  - Desestimar una transacción equivale a vaciar ese segmento.
  - Perpetrar una transacción equivale a ejecutar de modo atómico ese segmento sobre la BD.
  - A medida que se definen las operaciones de la transacción, se bloquean los recursos afectados con los correspondientes **cerrojos**
  - Los cerrojos se eliminan al finalizar la transacción.

Tema 8.6: Gestión transaccional en PL/SQL

- **Transacción**: conjunto de instrucciones de **actualización** que deben ser llevadas a cabo de modo atómico (como conjunto, "o todo o nada")
- Autocommit: define que todas las transacciones son mono-instrucción.
- Instrucciones: COMMIT (realizar) y ROLLBACK (deshacer)
  - `COMMIT [WORK]`
  - `ROLLBACK [WORK] [TO [SAVEPOINT] <savepoint>]`
  - `SAVEPOINT <savepoint>`



Tema 8.6: Cerrojos en Oracle

- Los cerrojos permiten algunas operaciones y otras no.
- Si una operación necesita acceder a un recurso bloqueado (para ese tipo de operación) deberá esperar a que se libere (si la operación lleva la opción NOWAIT devuelve inmediatamente el control con un error).
- También puede esperar una cantidad de tiempo dada en segundos.

- **Cerrojo de datos** (DML Lock): bloquea datos (nivel de tabla o de fila). Pueden ser automáticos o creados por el usuario.

- Cerrojo de Catálogo (DDL Lock): bloquea la estructura (tablas o vistas) para evitar cambios DDL sobre una estructura que está siendo operada

- Cerrojo interno (*latch*): bloquea estructuras internas de la BD (blocks). INITRANS (MAXTRANS) permiten definir el número de transacciones concurrentes (inicial/fijo, máximo/variable) en un cubo.

- Cerrojo distribuido: asegura la consistencia entre varias instancias de un servidor distribuido. En particular, un PCM (parallel ce management) es un bloqueo distribuido sobre uno o más bloques de datos (de tablas o de índices) en la memoria intermedia (buffer).

Tema 8.6: Cerrojos de Datos en Oracle

- Algunos cerrojos son de creación automática (RX), y todos ellos pueden crearse con la instrucción: `LOCK TABLE <tablename> IN <mode>`
- Todos los cerrojos permiten consultar (lectura consistente: en caso de que la tabla esté siendo modificada, se accede a la versión anterior).
- Contemplan dos niveles (tabla/fila) y dos modos (compartido/exclusivo)

fila

  - Row Share (**RS**): el menos restrictivo; impide que otra transacción obtenga un cerrojo exclusivo sobre determinadas filas. Se puede obtener con `SELECT… FOR UPDATE [OF column]` (*cerrojo de cursor*; con `OF column` afecta sólo a las tablas con atributos enlistados).
  - Row Exclusive (**RX**): como el (X) pero sólo sobre determinadas filas (Oracle aplica este por defecto durante INSERT/UPDATE/DELETE).

tabla

  - Share (**S**): bloquea una tabla, impidiendo que otras transacciones obtengan un cerrojo exclusivo (impide actualizaciones).
  - Share Row Exclusive (**SRX**): idem, pero además impide (S) a otros
  - Exclusive (**X**): es el más restrictivo; bloquea todo (menos la query).

Tema 8.6: Compatibilidad de cerrojos

a user has… \ other users can…	X	SRX	S	RX	RS
RS	✗	✓	✓	✓	✓
RX	✗	✗	✗	✓	✓
S	✗	✗	✓	✗	✓
SRX	✗	✗	✗	✗	✓
X	✗	✗	✗	✗	✗

* sobre filas diferentes

- Algunos DBMS escalan cerrojos (de fila a tabla), pero Oracle no.
- ¡Cuidado con los cerrojos restrictivos en transacciones largas!
- Interbloqueo: dos transacciones bloqueando sendos recursos y tratando de bloquear el otro recurso quedan interbloqueadas (deadlock). Para evitarlo, Oracle resuelve cancelar una de ellas (por marcas de tiempo).
- En transacciones distribuidas, Oracle puede confundir los interbloqueos con bloqueos largos (resuelve con un temporizador en trans. distribuidas).